

Skinner et al.

S/N: 09/678,207

### REMARKS

Claims 1-37 are pending in the present application. In the Office Action mailed March 28, 2003, the Examiner rejected claims 1, 18-19, and 31-33 under 35 U.S.C. §103(a) as being unpatentable over Admitted Prior Art (APA) in view of Davidson et al. (USP 5,630,136). The Examiner next rejected claims 5-14 under 35 U.S.C. §103(a) as being unpatentable over APA in view of Davidson et al., Orton et al. (USP 5,475,845), Lee (Adding External Input Sources to the X Toolkit Event Loop), and further in view of Connelly et al. (USP 5,706,515).

#### Objection to Drawing

The Examiner objected to the drawings because Fig. 1 does not include the reference number "29," as described in the Detailed Description. Applicant appreciates the thorough examination and has amended the Specification to remove reference number "29." As such, Applicant believes the drawings fully comply with 37 CFR 1.84(p)(5).

#### Background

Before addressing the Examiner's rejection with respect to the specific elements of the claims, Applicant believes it would be helpful to highlight a fundamental misunderstanding upon which Applicant believes the current rejection is based. Specifically, Applicant believes the Examiner has underappreciated the many unique circumstances and challenges presented when combining applications written in Java with the X Windows visualization toolkit. Once these challenges are appreciated, the unique solution of the claimed invention will become readily apparent.

Java is a somewhat unique programming language in that it has the ability to be platform independent. That is, programs written in Java can run on a computer regardless of the specific operating system that is present on a given computer. Unlike traditional object-oriented programming languages that are compiled for a specific platform or operating system, Java is compiled into bytecodes. These bytecodes are specific to the Java Virtual Machine (JVM) but not to any particular platform or operating system. Therefore, as long as a JVM exists for a particular platform or operating system, the Java program can be run on that particular platform.

While the fact that Java programs are not platform specific is typically of great advantage, it does present challenges when Java programs need to interact with programs

Skinner et al.

S/N: 09/678,207

compiled in a platform specific programming language. As stated in Applicant's Specification, the X Window visualization toolkit, which is also referred to as X Toolkit Intrinsics or Xt Intrinsics, is written in the C or C++ programming languages. See Pg. 3, lns. 3-5 and pg. 13, lns. 11-13. However, Java does not provide direct support for programs written in C or C++. See Applicant's Specification, pg. 13, lns. 20-24. Therefore, when a Java program needs to interact with the X Window visualization toolkit there are multiple challenges that are presented. Some of these challenges stem from (1) the limitations of the X Window visualization toolkit itself and some stem from (2) the fact that the X Window visualization toolkit is written in C or C++.

First, Applicant's Background, Davidson et al., and the Examiner all recognize and acknowledge that the X Window visualization toolkit is not capable of handling simultaneous access by multiple threads of a multithreaded program. One solution to this problem is serialization of thread access. As stated by the Examiner with respect to claim 1, "Davidson...provides a technique for serializing access to multithreading unsafe resource[s]." Office Action of March 3, 2003, pg 3. That is, Davidson et al. recognizes that the X Window visualization toolkit is a multithreading unsafe resource and therefore, access by a multithread program must be serialized, or else risk corruption.

However, the Examiner has overlooked, and Davidson et al. does not consider, the second category of challenges. That is, specific considerations are required before a Java program can use existing native applications and native libraries written in C or C++, such as the X Window visualization toolkit. Simply, Java does not provide direct support for Xt Intrinsics because it is typically written in C or C++. See Applicant's Specification, pg. 13, lns. 20-24. As such, the current invention provides the novel ability to allow the Java program and the C-based X Window visualization toolkit to interact properly. See Applicant's Specification, pg. 13, ln 19 to pg. 20, ln. 6 and Fig. 4. On the other hand, neither the APA nor Davidson et al. teaches any such ability. In fact, Davidson et al. does not address Java at all.

#### Rejection of Claims 1, 18, 19, and 31-33

The Examiner rejected claims 1, 18-19, and 31-33 under §103(a) as being unpatentable over APA in view of Davidson et al. To establish a *prima facie* case of obviousness, the Examiner must show (I) there is a suggestion, or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the

Skinner et al.

S/N: 09/678,207

art to modify the reference or combine the reference teachings, (II) a reasonable expectation of success, and (III) the references must teach or suggest all the claim limitations. MPEP §2142. As will be explained, the Examiner has failed to provide the requisite (I) motivation to combine, has not shown a (II) reasonable likelihood of success, and failed to show that (III) each and every claimed limitation is either taught or suggested.

When referring to claims 1 and 18, the Examiner recognizes that "APA does not explicitly teach suspending execution of the X event loop to prevent concurrency related data corruption while a call to the X Window visualization toolkit is made by the Java application thread." Office Action of March 3, 2003, pg. 3. The Examiner continues by stating that "Davidson teaches only one thread is permitted to access the resource." *Id.* Finally, the Examiner asserts that "[i]t would have been obvious to apply the teaching of Davidson to the system of APA because it provides a technique for serializing access to multithreading unsafe resource." Applicant respectfully disagrees with the Examiner's assertion.

With regard to claim 18, the Examiner added that "[i]t is well know in the art that a Java application is a multi-thread application, one of ordinary skill in the art would be able to modify the Java application to have multiple threads each makes a call to the X Window toolkit." The Examiner also stated that "Davidson teaches only one thread could access the resource, and the rest of the threads are suspended."

Applicant does not disagree that it is well known that a Java Application may be threaded and Applicant does not necessarily disagree that Davidson et al. permits a single thread access to a resource and suspends the remaining threads of a multi-thread program. However, this is the very reason that Applicant provided the APA in the Application in the first place.

It is important to realize that the claimed invention provides a novel solution to not one, but to two challenges. First, the claimed invention provides a means for a Java application to directly communicate with the C or C++ based X Window visualization toolkit. As stated in the APA, previous methods of direct communications, i.e. a bridge, between a Java application and the X Window visualization toolkit have been to separate the Java application into a client portion and server portion. However, the current client/server solution raises yet a second challenge that is solved by the current invention. Specifically, the X Window visualization toolkit is a multithread unsafe resource and the

Skinner et al.

S/N: 09/678,207

client/server solution for direct communication does not afford the ability to restrict access to the X Window visualization toolkit. As will be explained, while Davidson et al. may teach a method of serializing access to multithread unsafe resources, it does not teach or suggest a resolution to the challenges of incorporating the unique Java programming language and, therefore, Davidson et al. does not teach a system or method operable within the conditions exclusive to Java. Simply, while the APA provides a partial solution to the first challenge and Davidson et al. provides a partial solution to the second challenge as will be explained in detail, the solutions are incompatible and therefore, do not present, or even suggest, a solution to both challenges that can be used simultaneously. Specifically, a combination of the APA and the serialization of Davidson et al. would not be operable because the client/server solution of the APA cannot be readily serialized by the method of Davidson et al. Therefore, a rejection based on this combination fails to take into account the challenges presented by implementations incorporating Java and the combination of its multithreading and inability to directly communicate with the X Window visualization toolkit.

One of ordinary skill in the art will readily recognize that (I) Davidson et al. does not teach or suggest a method for serializing access of Java threads to multithread unsafe resources, (II) the combination of Davidson et al. and the APA is not likely to succeed, and (III) the combination of Davidson et al. and the APA does not teach each and every element of the claimed invention.

#### I. Motivation to Combine

One of ordinary skill in the art looking to create a method for serializing access of Java threads to multithread unsafe resources would not consider the teachings of Davidson et al. because Davidson et al. does not teach the use of Java at all.

As stated, Java does not directly support programs written in C or C++. Therefore, it is necessary to create a bridge between the Java and the C or C++ to facilitate communications. While Davidson et al. describes conceptually how a baton manager would behave within an X Window framework, Davidson et al. does not teach implementing a baton manager to bridge C++ and Java. Davidson et al. does not address Java at all and is specific that "[t]he baton processing 50 is executed on a computer system and operating system that supports multithreading." Col. 5, ln. 66 to col. 6, ln. 1. Therefore, Davidson et al.

Skinner et al.

S/N: 09/678,207

is clear that the baton manager is executed solely within the operating system and provides no teaching or suggestion for the interaction of the baton manager with any other resources such as a JVM. One of ordinary skill in the art will readily recognize that the baton manager of Davidson et al. is incapable of bridging the C or C++ based X Window visualization toolkit and a Java program or a JVM because Davidson et al. makes no considerations for the specific requirements of Java. This is an important distinction that cannot be overlooked. Simply, in order to create a method for serializing access of Java threads to multithread unsafe resources any baton that is implemented must bridge C++ and Java to support Java control over X Intrinsics implemented graphics. As previously stated, this requirement stems from the nature of Java as a portable programming language and its relationship to C or C++ as programming languages. Specifically, Java does not provide direct support for Xt Intrinsics because it is typically written in C or C++.

The current invention provides a unique method of overcoming this challenge. Specifically, claim 1 calls for "providing a JAVA application thread that includes a call to an X Window visualization toolkit and a JAVA process thread that comprises an X Window X event loop and suspending execution of the X event loop to prevent concurrency related data corruption while a call to the X Window visualization toolkit is made by the JAVA application thread." While the novel method of the claimed invention affords a means for the Java program to directly communicate with the X Window visualization toolkit without the risk of corruption, neither Davidson et al. nor the APA provides any teaching or suggestion of such a method.

Therefore, one of ordinary skill in the art would not be motivated to combine the APA and Davidson et al. in an attempt to create a method for serializing access of Java threads to an X Window visualization toolkit because neither the APA nor Davidson et al. teaches a baton manager that is also capable of bridging C or C++ and Java. Since the communication between the Java program and the C or C++ based X Window visualization toolkit is paramount to the feasibility of the method, one of ordinary skill in the art would not even consider Davidson et al. when looking to create the claimed method.

Skinner et al.

S/N: 09/678,207

## II. Likelihood of Success

Furthermore, one of ordinary skill in the art would not reasonably expect that the combination of the APA and Davidson et al. would render the claimed method obvious because, as previously stated, neither teaches the ability of a Java program to communicate with the C or C++ based X Window visualization toolkit. Simply, a combination of the APA and Davidson et al. would be unworkable as a method of integrating an X Window visualization toolkit with a Java application because Java does not provide direct support for communication with C or C++ based programs. One of ordinary skill in the art will readily recognize that a method of serializing the access of Java threads to the X Window visualization toolkit would be unworkable without a bridge to permit the Java program to communicate with the X Window visualization toolkit. Any method of serializing the access of Java threads to the X Window visualization toolkit requires the specific teachings of the current invention, none of which are taught by the art of record. That is, claim 1 calls for providing a JAVA application thread that includes a call to an X Window visualization toolkit and a JAVA process thread that comprises an X Window X event loop and suspending execution of the X event loop to prevent concurrency related data corruption while a call to the X Window visualization toolkit is made by the JAVA application thread" and no such method is taught or suggested by the APA or Davidson et al. Therefore, there is no reasonable likelihood of success of obtaining the invention by combining the art of record.

## III. Elements of the Claim

Also, the combination fails to teach or suggest each and every element of the claims. Referring to claim 1, the Examiner asserts that the APA teaches "providing a JAVA application thread that includes a call to an X Window visualization toolkit (Java application requests a schedule camera position...is rotated), and a JAVA process thread that comprises an X Window X event loop (X event loop)." The Examiner is citing page 4 of Applicant's Specification. However, the paragraph before the Examiner's citation clearly states that "[p]ast and present solutions to reusing Xt Intrinsics based visualization toolkits within a JAVA application or applet require that the application be separated into client and server processes and have some form of interprocess communication." Therefore, the APA does not teach "a JAVA process thread that comprises an X Window X event loop" but instead teaches separating the application into client and server processes. One of ordinary skill in the art will readily recognize that separating the application into client and server processes is

Skinner et al.

S/N: 09/678,207

not the same as dedicating a Java thread to an X event loop. Furthermore, Davidson et al. does not teach the use of Java at all. As such, the combination does not teach each and every element of the claim. Since elements are completely absent in the prior art, Applicant believes the rejection is not sustainable.

Therefore, (I) the combination of the APA and Davidson et al. does not teach or suggest a method for serializing access of Java threads to an X Window visualization toolkit, (II) one of ordinary skill in the art would not have a reasonable expectation of success when combining Davidson et al. and the APA and (III) the combination does not teach each and every element of the claimed invention. Simply, neither the APA nor Davidson et al. teaches a method for serializing access of Java threads to a X Window visualization toolkit, which is a C or C++ program. As previously shown, the current invention provides a specific implementation to serialize access to the X Window visualization toolkit and allow the Java program to directly communicate with the C or C++ based X Window visualization toolkit. The current invention provides a unique solution to a problem that is unique to providing Java applications with access to an X Window visualization toolkit, in that only Java presents the specialized set of circumstances that compound both of the challenges solved by the current invention. Accordingly, Applicant believes that claims 1 and 18 are patentably distinct from the art of record. Furthermore, claims 19 and 31-33 are in condition for allowance pursuant to the chain of dependency.

Rejection of Claims 3, 4, 21, 22, 24-30, and 34-37

The Examiner rejected 3, 4, 21, 22, 24-30, and 34-37 under §103(a) as being unpatentable over the APA in view of Davidson et al. and further in view of Orton et al. and Lee (Adding External Input Sources to the X Toolkit Event Loop). Regarding claim 3, 4, 21, 22, and 24-30, Applicant believes the claims are in condition for allowance pursuant to the chain of dependency, as the claims depend from allowable claims 1 and 18.

Regarding claims 34 and 35, the Examiner rejected the claims by simply referring Applicant to the previous rejections of claims 1 and 3. As previously shown, the Examiner has failed to establish a *prima facie* case of obviousness with respect to claim 1. Conversely, Applicant has shown that claim 1 is patentably distinct from the art of record. Furthermore, Applicant has shown that claim 3 is in condition for allowance pursuant to the chain of dependency. Therefore, since the rejection of claims 1 and 3 is unsustainable, the

Skinner et al.

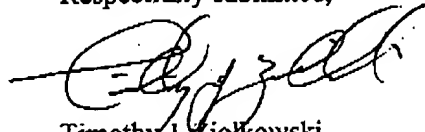
S/N: 09/678,207

rejection of claims 34 and 35 on the same basis as claims 1 and 3 is also unsustainable. As such, claims 34 and 35 are believed patentably distinct from the art of record. Accordingly, claims 36 and 37 are in condition for allowance pursuant to the chain of dependency.

Therefore, in light of the foregoing clarifications, Applicant respectfully believes that the present application is in condition for allowance. As a result, Applicant respectfully requests timely issuance of a Notice of Allowance for claims 1-37.

Applicant appreciates the Examiner's consideration of these Amendments and Remarks and cordially invites the Examiner to call the undersigned, should the Examiner consider any matters unresolved.

Respectfully submitted,



Timothy J. Ziolkowski  
Registration No. 38,368  
Direct Dial 262-376-5139  
[tjz@zpspatents.com](mailto:tjz@zpspatents.com)

Dated: June 9, 2003  
Attorney Docket No.: GEMS8081.029

**P.O. ADDRESS:**  
Ziolkowski Patent Solutions Group, LLC  
14135 North Cedarburg Road  
Mequon, WI 53097-1416  
262-376-5170